Unit 1: PythonによるTI-Innovator™ Hubのスタート

このレッスンでは，Pythonプログラムの作成・実行と，TI-Innovator Hubのlight(赤色LED)を使った基本事項を学習します。

Skill Builder 1: ライトアップ

目標

- Pythonプログラムの作成・実行
- TI-Innovator Hubのlightを制御

Teacher Tip: このコースでは，学生がプログラミング経験がないことを前提として，TI-Nspire CX IIにおいてPythonプログラミングを使ったTI-Innovator Hubを紹介します。Pythonプログラミングのより直接的な導入については，education.ti.com のTI Codeで入手できるTI-Nspire CX II Pythonプログラミングに関するコースを参照してください。

TI-Nspire CX II上のPythonは，Pythonのサブセットであるマイクロパイソンを使って実装されています。2つの重要なことがあります。

1. MicroPythonには，Buit-in(組み込み)やstandard(標準)機能であっても，標準Python関数がすべて含まれる訳ではありません。たとえば，shuffle()はlist()関数の一部ではありません。
2. MicroPythonモジュールや関数は，すべてTI-Nspire CX IIメニューにある訳ではありません。最も一般的に使われる関数がメニューにあります。

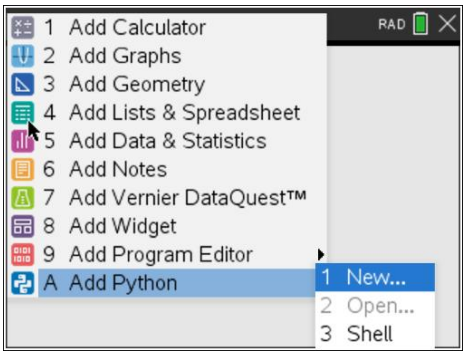
TI-Innovator Hubを使うには，Pythonプログラムがti_hubモジュール**from ti_hub import ***をインポートする必要があります。Rover(ユニット4, 5, 6)の場合には，**import ti_rover as rv**です。よって，すべてのRoverステートメントは，”rv”で始まります。TI-RGB Array(ユニット7)の場合には，from ti_hub import *と，RGB Arrayクラスのインスタンスを作成します。*var*=**rgb_array()**。

このコースのはじめのユニット1, 2, 3では，Hubの組み込みデバイスである**LIGHT**(光)，**COLOR**(色)，**SOUND**(音)，**BRIGHTNESS**(明るさ)センサに焦点を当てています。次の3つのユニット4, 5, 6ではRoverを扱い，最後のユニット7ではTI-RGB Arrayを扱います。Texas Instrumentsは，TI-Nspire CX II，TI-Innovator Hub，TI-Innovator™ Roverをサポートするため，’ti_’Pythonモジュールをすべて作成しました。TI-Innovator HubとRoverモジュールは，コーディングにオブジェクト指向のアプローチを使っているため，コマンドと構文はTI-Basicや他のPythonコースとは大きく異なります。

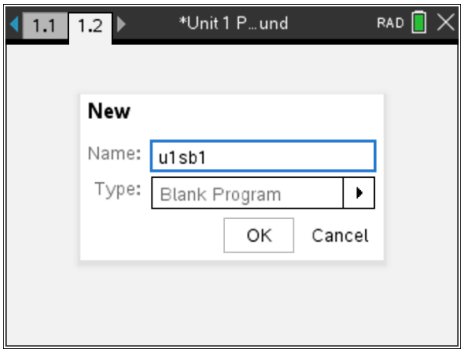
TI-Nspire CX IIでPythonを使ったTI-Innovator Hubプログラミングの世界へようこそ!

最初のプログラムは，TI-Innovator Hub回路基板の赤色LEDを操作します。ボード上では見づらいですが，電源を入れると分かります。

1. 新規TI-Nspire™ドキュメントを開きます。利用可能なアプリケーションが一覧表示されます。**Add Python**を選択してから，**New…**を選択します。

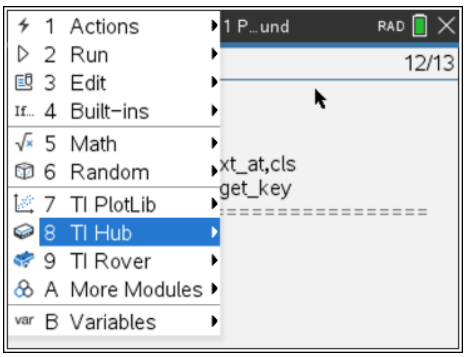


2. Pythonプログラム名として**u1sb1**を入力し，**enter**を押します。
   2番目の欄のTypeについては，後で説明します。



Teacher Tip: Pythonプログラム(ファイル)はTI-Nspireドキュメントの一部です。メニューから**Open…**を選択すると，ドキュメント内のPythonファイル(存在する場合)が表示されます。
次のステップでは，TI-Innovator Hubを制御するため必要な関数をインポートします。
Pythonは設計上，軽量です。多くの堅牢なプログラムを作成するために必要なのは，「組み込み」だけです。 Pythonにさらにパワーを与えるため，他のモジュールが利用可能です。 TI開発者は，PythonでのTI-Nspire CX IIのプログラミングを豊かで楽しいものにするカスタムモジュールを作成しました。

3. これでPythonエディタが表示されます。**menu**キーを押します。各メニュー項目には，関連するPythonプログラミングツールが表示されています。今のところ，私たちの主な関心は**TI Hub**メニューです。そのメニューから一番上の項目を選択します。



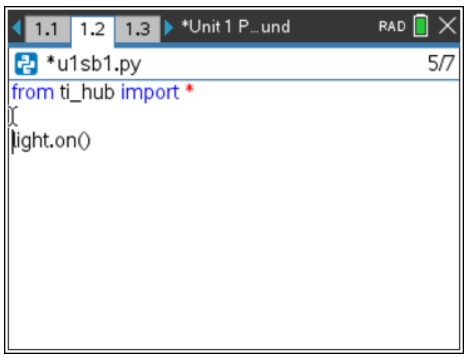
from ti_hub import *

このPythonコマンドは，TI-Innovator Hubでデバイスを操作する(または接続する)ため必要なツール(コマンド)を提供します。

また，このステートメントはTI-Innovatr Hubが接続されているかどうかを確認します。そうでない場合，プログラムは実行されません。

Teacher Tip: ti_hubモジュールには，TI-Innovator Hubが存在するかどうかを確認する機能が含まれています。

4. 使用するステートメントは次のものです。



light.on()

これが何をするのか予測できますか。

このステートメントは，次のところにあります。

menu > TI Hub > Hub Built-in Devices > Light Output > on()
(メニュー>TI Hub>Hub内蔵デバイス>光出力>オン())

Hub関係の関数は，すべてTI Hubメニューにあります。

Teacher Tip: 文字列の入力はほとんど必要ありません。コーディングの多くはメニューから直接，選択できます。

繰り返し：TI-Innovator Hubを接続せずにこのプログラムを実行しようとすると，エラーメッセージが表示され，プログラムが停止します。**import ti_hub**ステートメントは，TI-Innovator Hubが利用できるかどうかをチェックします。

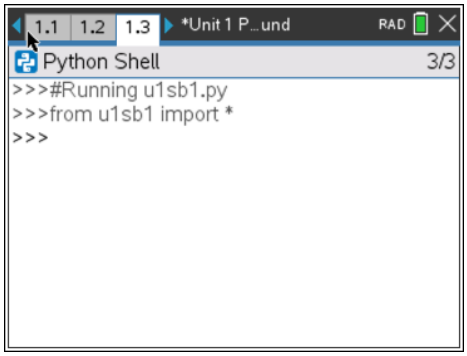
初心者以外：TI-Innovator Hubを接続せずにTI-Innovator Hubプログラムを実行できるようにするには(他のコーディングの問題をチェックするため)，try…except構造を使います。

```
try:
    from ti_hub import *
    hub = True
except:
    hub = False
```

そして，プログラム全体を通してHub命令が呼び出されるたびに，最初にHubの状態をチェックします。

```
if hub: light.on()
```

5. これで，この素晴らしいプログラムを実行する準備が整いました。使うことができます。



menu > Run > Run (Ctrl+R)

(メニュー>実行>実行(Ctrl＋R))

実行には，ショーカットキー**ctrl+R**と押すだけでもかまいません。TI-Nspire CX IIの画面は右図のようになります。**ctrl+R**を押すとドキュメントにページが追加され，Python Shell(シェル)アプリが開きます。右図はPythonシェル画面です。シェルはTI-Nspireの電卓アプリケーションに似ています。これは，Pythonプログラムが実行される場所です(実際には，唯一の場所です)。

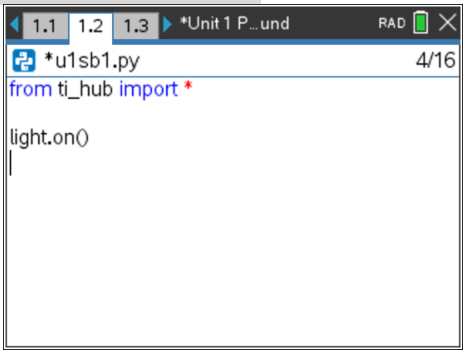
記号>>>はPythonコマンドプロンプトです。次のコマンドを待っています。しかし，TI-Innovator Hubを見ると，赤色LEDが点灯しているのが分かります。これは，プログラムの**light.on()**ステートメントの結果です。

Teacher Tip: light.on()は見た目よりもはるかに複雑なステートメントです。lightは**ti_hub**モジュール内で作成されたクラスのインスタンスです。.on()はライトをオンにするクラスのメソッドです。表面的には簡単に使えるコマンドですが，その下では多くの作業が行われています。これがPythonの機能です。これによりまったく

の初心者でも具体的なプログラミングにアクセスできます。

Note: クラス(class)：文字型や数値型など様々な型の種類をクラスといいます。
インスタンス(instance)：型の実体のことです。
メソッド(method)：メソッドも関数ですが，関数が単独で呼び出しできるのに対して，メソッドは変数や値に付けて呼び出します。

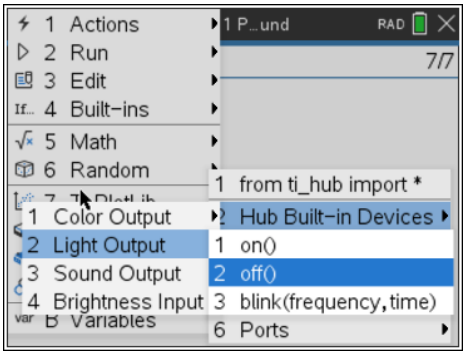
6. Pythonプログラムは，シェルページの前のページにあります。プログラム編集に戻るには，**ctrl+左矢印**を押します(右図)。TI-Innovator Hubで，赤色LEDがまだ点灯していることに注意します。



7. どのコマンドがライトをオフにするか推測できますか。それは次のところから見つけることができます。

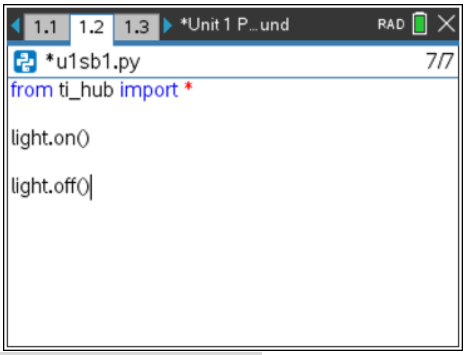
   menu > TI Hub > Hub Built-in Devices > Light Output > …

   (メニュー>TI Hub>Hub内蔵デバイス>光出力>...)



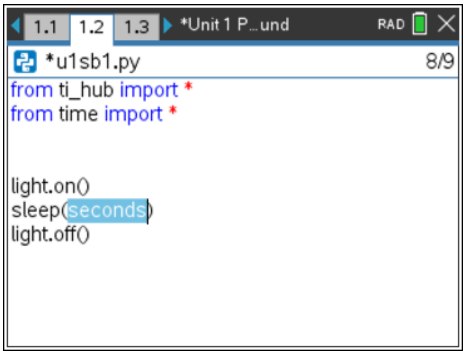
8. **light.on()**コマンドの後に**light.off()**コマンドを追加します。行を空けてもかまいません。これらは実行に影響を与えず，コードを読みやすくします。プログラムを再度実行してください。LEDが速く点滅するのが分かりますか。速すぎる?

   次のいくつかのステップでは，LEDが点灯している時間を制御する機能を追加します。



Teacher Tip: 別のimport(インポート)コマンドがあります。後のレッスンでは，TI-Innovator Hubプログラミングに必要なブロックを提供するプログラミングプロジェクトtemplateを使います。

9. **from ti_hub import ***の下の行にカーソルを置きます。



menu > More Modules > Time（メニュー>その他のモジュール>時間）と押して，次を選択します。

from time import *

light.on()と**light.off()**の間に，次のステートメントを追加します。

sleep(seconds)

これは，**Time**メニューと同じところにあります。

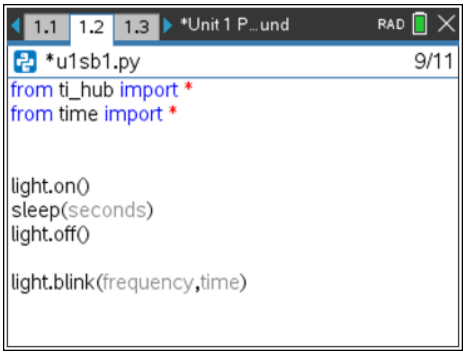
seconds(秒)はプレースホルダー(仮に置いた文字列)です。

2や3などの数字に置き換えます。

sleep()関数は，プログラムの次のステートメントに進む前に，その秒数の間，待機または一時停止するようコンピュータに指示します。

ここでプログラムを実行すると(**ctrl+R**を押す)，LEDはオフになる前に選択した秒数の間，点灯したままになります。

10. ライトを点滅させるには，プログラム内の一連のステートメントを繰り返すか，または…。



Light Output(光出力)メニューには，LEDを点滅させる**blink()**関数もあります。**light.blink()**には，frequency(周波数，頻度)とtime(時間)の2つのパラメータがあります。両方を数値に置き換えて，パターンを理解してください。ポップアップツールチップに注意します!

Teacher Tip: frequency(周波数，頻度)が選択されている場合，**blink()**関数にはツールチップ0.1～20Hzが表示されます。Hzはヘルツを表し，1秒あたりのサイクル数を意味します。したがって，frequencyが3の場合，LEDは1秒間に3回点滅します。timeパラメータ(0.1～100秒)は，TI-Innovator Hubに点滅する秒数を指示します。したがって，これは単純なかけ算です。light.blink(3, 5)はLEDを5秒間に3*5＝15回点滅させます。

繰り返し：**.on()**，**.off()**，**.blink()**はすべてlight変数がインスタンスであるlightクラスのメソッドです。これはオブジェクト指向プログラミング(Object-Oriented Programming，OOP)であり，TI-Innovator HubやRoverツールはその中で設計されています。